This Listing of Claims will replace all prior versions, and listings, of claims in the application.

LISTING OF CLAIMS:

1. (Currently Amended) A floating point execution unit for performing multiply/add operations on a floating point number comprising a plurality of operands taken from an instruction having a plurality of floating point number operand positions, wherein the performing does not implement operand formatting selection and unpacking, thereby increasing a speed at which the floating point number is output, the floating point unit comprising:

a multiplier for calculating a product of two of the operands;

an aligner directly coupled to the multiplier for aligning said product and a third of the operands in a first data path;

wherein the [[a]]first data path is for supplying to the multiplier operands from a first and a second of the operand positions of the instruction;

a second data path for supplying the third operand to the aligner; and

a multiplexer on the second data path for selecting, for use by the aligner, either the operand from the second operand position of the instruction or the operand from the third operand position of the instruction, and supplying same to the multiplier;

wherein the first data path is maintained free of multiplexer operations, thereby increasing a speed at which the floating point number is output.

2. (Cancelled)

3. (Previously Presented) A floating point execution unit according to Claim 1, wherein:

the aligner computes a shift amount for aligning said product and the third operand; and

the multiplexer operates to select the third operand as the aligner computes the shift amount.

4. (Cancelled)

5. (Previously Presented) A floating point execution unit according to Claim 3, wherein each of the operands and said product includes an exponent value, and aligner computes said shift amount based only on said exponent values.

6. (Previously Presented) A floating point execution unit according to Claim 1, wherein each of the operands has an exponent value, and floating point execution unit determines whether the exponent values of any of the operands is zero in parallel with operation of the multiplier and the aligner.

7. (Previously Presented) A floating point execution unit according to Claim 6, wherein said floating point execution unit tests said exponent values for a zero value while the multiplier calculates said product.

8. (Previously Presented) A floating point execution unit according to Claim 6, wherein the floating point execution unit establishes a test result number based on results of said determination.

9. (Cancelled)

10. (Previously Presented) A floating point execution unit according to Claim 8, wherein the plurality of bits are used to force special values into the aligner result.

11. (Previously Presented) A floating point execution unit according to Claim 3, wherein the aligner compresses two of the three input exponents and an offset while selecting the third exponent.

12. (Previously Presented) A floating point execution unit according to Claim 11, wherein, when executing an add or subtract instruction, the aligner computes the alignment shift amount as: [exponent ea + exponent eb – exponent 2eb].

13. (Currently Amended) A method of operating a floating point execution unit to perform multiply/add operations on a floating point number without implementing operand formatting selection and unpacking thereby increasing a speed at which the floating point execution unit operates, the floating point unit having a multiplier, an aligner directly coupled to the multiplier in a first data path, and a multiplexer, the method comprising ~~the~~ steps:

sending an instruction to the floating point unit, including a floating point number upon which multiply/add operations are to be performed, the instruction having a plurality of operand positions holding operands of the floating point number;

using the multiplier to calculate a product of two of the operands of the instruction;

using the aligner to align said product and a third of the operands of the instruction;

supplying over the [[a]]first data path to the multiplier operands from a first and a second of

the operand positions of the instruction, wherein said first data path is free of multiplexer operations;

supplying over a second data path a third operand of the instruction to the aligner;

positioning the multiplexer on the second data path;

using the multiplexer to select, for use by the aligner, either the operand from the second operand position or the operand from the third operand position; ~~and~~

outputting the selected operands to the aligner; and

outputting the floating point number upon which the multiply/add operations were performed;

wherein the first data path is maintained free of multiplexer operations, thereby increasing a speed at which the floating point number is output.

14. (Cancelled)

15. (Original) A method according to Claim 13, comprising the further step of:

using the aligner to compute a shift amount for aligning said product and the third operand; and wherein the multiplexer operates to select the third operand in parallel with the aligner.

16. (Original) A method according to Claim 15, wherein the multiplexer selects the third operand while the aligner computes said shift amount.

17. (Original) A method according to Claim 15, wherein each of the operands and said product includes an exponent value, and the step of using the aligner to compute said shift amount includes the step of computing said shift amount based only on said exponent values.

18. (Original) A method according to Claim 13, wherein each of the operands has an exponent value, and comprising the further step of, determining, in parallel with the multiplier and the aligner, whether the exponent values of any of the operands is zero.

19. (Original) A method according to Claim 18, wherein the step of determining whether the exponent values of any of the operands is zero occurs while the multiplier calculates said product.

20. (Original) A method according to Claim 18, comprising the further steps of:

establishing a test result number based on results of said determination, the test result number including a plurality of bits;

using a first of the bits to indicate whether the addend is zero; and

using a second of the bits to indicate whether the product is zero.